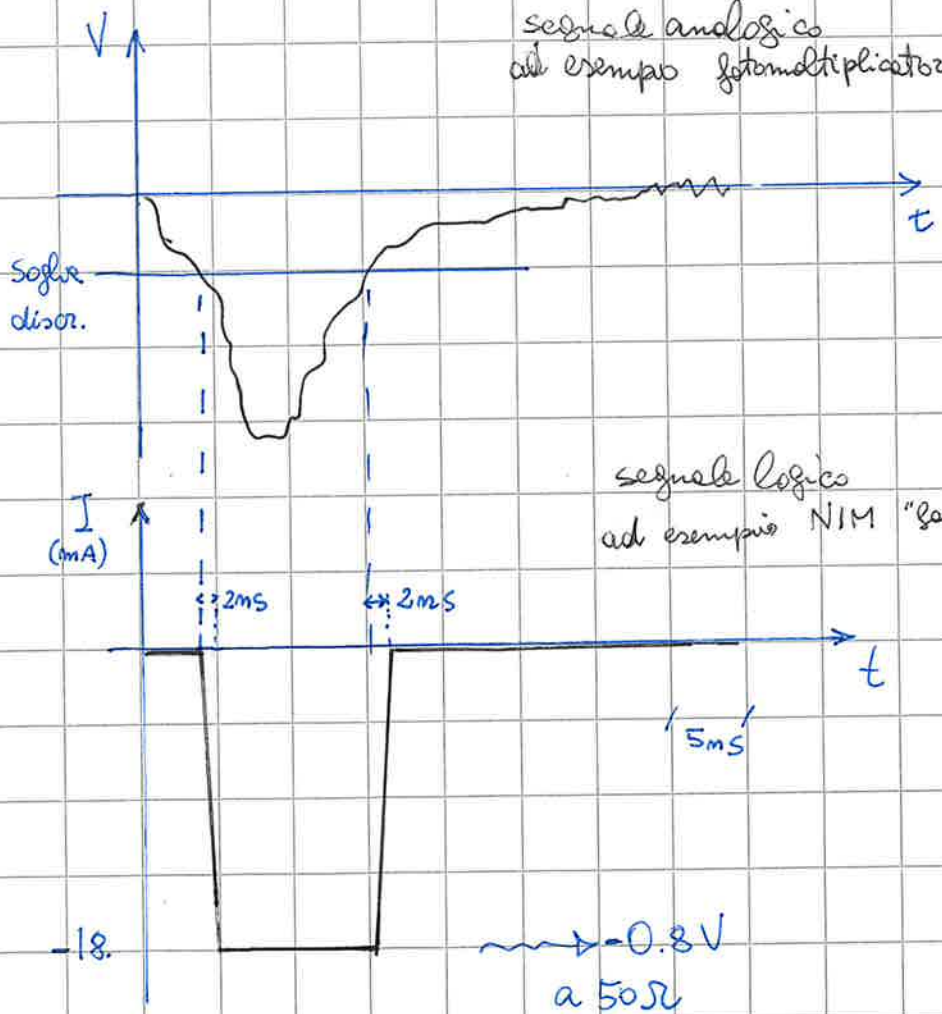
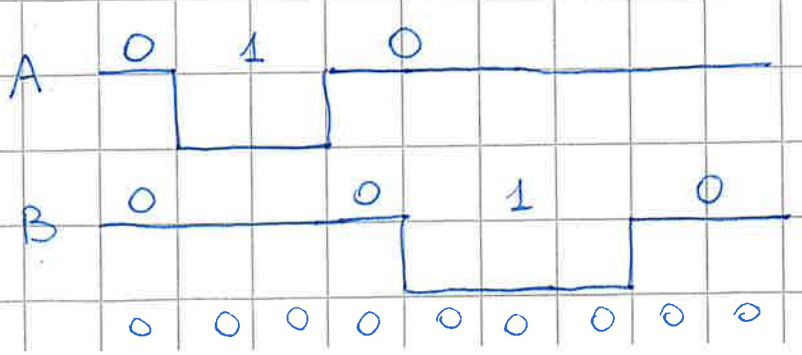
Mercoledì 14 aprile 2021 ore 11 | aggiornate Lunedì 6 marzo 2023 ore 9 ①

• BOOLEAN LOGIC & ch. 16 LEO
Electronic Logic for Experiments

Prendiamo un segnale analogico da un rivelatore
lo trasformiamo in uno <u>logico</u> → discriminatore
→ ADC

e poi facciamo analisi semplici tipo conteggi,
ma sotto certe condizioni ad es se in coincidenza
con un altro segnale (AND gate)
→ questo è un primo esempio di logica Booleana.

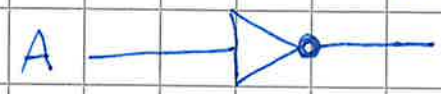




Questi due non sono mai in coincidenza: mai nello stesso stato logico allo stesso intervallo di tempo

Basic logic Gates: Symbols

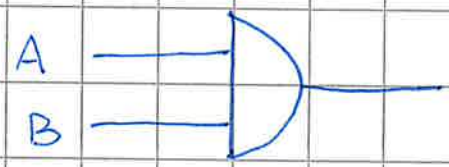
In laboratorio useremo moduli (al cui interno i circuiti che) eseguono elettronicamente l'equivalente delle operazioni Booleane su UNO o DUE segnali in ingresso (INPUT SIGNALS) A, B producendo in uscita (OUTPUT of the GATE) il segnale C.

GATE	INPUT	SIMBOLO	OUTPUT
NOT	A		C

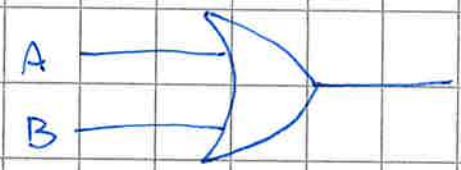
(è la più semplice operazione; matematicamente si indica con una "barra" sopra $\bar{A}$)

TABELLA DI VERITÀ

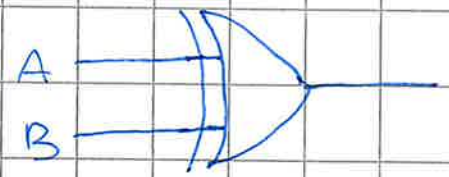
A	C
1	0
0	1

GATE	INPUT	SIMBOLO	OUTPUT
AND	A, B		C

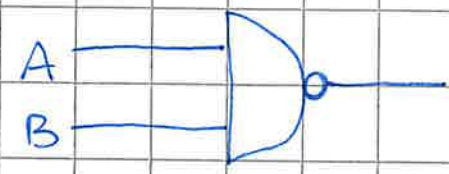
A	B	C=AB
1	1	1
1	0	0
0	1	0
0	0	0

GATE	INPUT	SIMBOLO	OUTPUT
OR (INCLUSIVE)	A, B		C

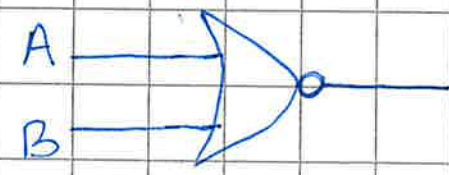
A	B	C=A+B
1	1	1
1	0	1
0	1	1
0	0	0

GATE	INPUT	SIMBOLO	OUTPUT
XOR (EXCLUSIVE)	A, B		C

A	B	C=A⊕B
1	1	0
1	0	1
0	1	1
0	0	0

GATE	INPUT	SIMBOLO	OUTPUT
NAND	A, B		C

A	B	$C=\overline{AB}$
1	1	0
1	0	1
0	1	1
0	0	1

GATE	INPUT	SIMBOLO	OUTPUT
NOR	A, B		C

A	B	$C=\overline{A+B}$
1	1	0
1	0	0
0	1	0
0	0	1

Matematicamente AND è simile a una moltiplicazione (si scrive A·B o AB)

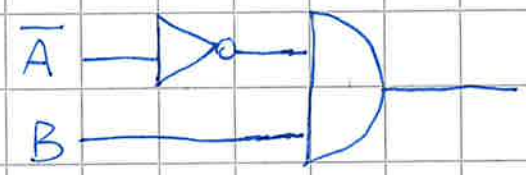
OR " somma (" " A+B)

Nell'esempio di fine pag (1) della coincidenza temporale, i segnali non erano mai in coincidenza. AND era sempre 0.

Ma se in sovrapposizione dovrei decidere il tempo minimo per poter vedere la coincidenza.

Nel modulo di coincidenza la sovrapposizione di due segnali con stato 1 è di almeno 5 ns

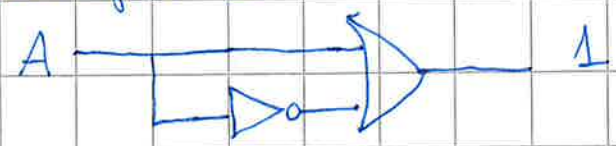
Potrei ad esempio negare il segnale A e allora dopo 20 ns $\bar{A}$ e B sarebbero in coincidenza per 15 ns.
Negare A corrisponde fare quello che si chiama una <u>anticoincidenza</u> o <u>inhibit gate</u>



- Esempi di leggi di identità Booleana con "OR"

$A + 0 = A$
$A + 1 = 1$
$A + A = A$
$A + \bar{A} = 1$

quindi per avere sempre lo stato logico 1



- Esempi d'identità booleana con "AND"

$$A\ 0 = 0$$
$$A\ 1 = A$$
$$A\ A = A$$
$$A\ \bar{A} = 0$$

- Per le operazioni AND e OR valgono le LEGGI:

ASSOCIATIVA $(A+B)+C = A+(B+C)$

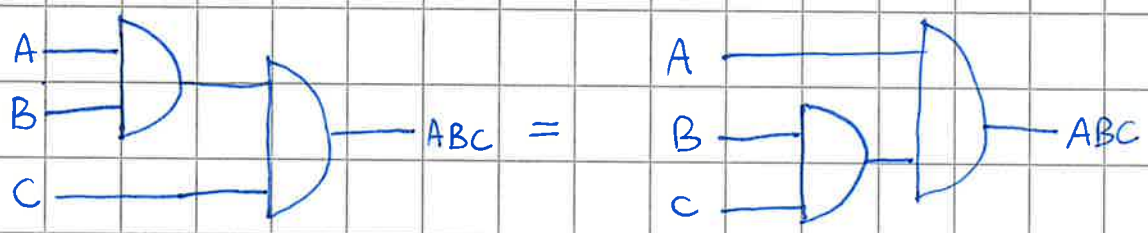
$(A\ B)\ C = A\ (B\ C)$ *

COMMUTATIVA $A + B = B + A$

$A\ B = B\ A$

DISTRIBUTIVA $A\ (B+C) = AB + AC$

ESEMPIO *



OSSERVAZIONE: nel trasferire questa logica Booleana nell'elettronica "pratica" le tre operazioni base AND, OR, NOT sono sufficienti, ma non tutte e tre necessarie, ad esempio per costruire tutte le altre operazioni logiche. Bastano { NOT e AND / NOT e OR

LEGGI di De Morgan:

1) $\overline{A\,B} = \overline{A} + \overline{B}$

2) $\overline{A + B} = \overline{A}\,\overline{B}$

generalizzabili a n variabili $\quad \overline{ABC\ldots} = \overline{A}+\overline{B}+\overline{C}+\ldots$

$\overline{A+B+C+\ldots} = \overline{A}\,\overline{B}\,\overline{C}\ldots$

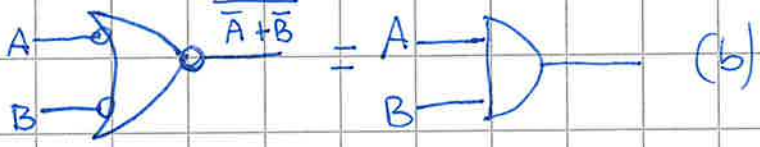
uso la 1) per "costruire" un "OR" solo con "AND" e "NOT"

$\overline{\overline{A}\,\overline{B}} = A+B$

(a)

oppure uso la 2) per costruire "AND" solo con "OR" e "NOT"

$\overline{\overline{A} + \overline{B}} = A\,B$



(b)

nel caso (a) ho costruito un "OR" con una coincidenza e NOT

" " (b) " " una coincidenza con "OR" e NOT

Ci "prendo gusto" e voglio ora costruire un "OR esclusivo" usando "OR" "AND" e "NOT" (me basterebbero due come visto sopra)

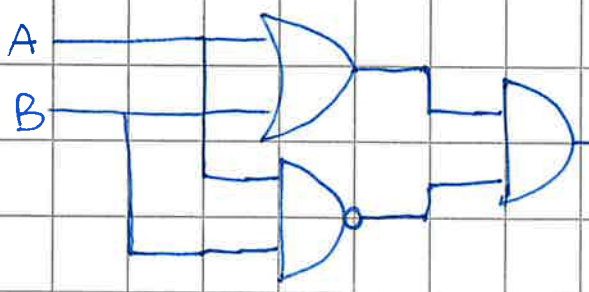
Algebricamente l'OR esclusivo si può scrivere come

$C = (A+B)(\overline{AB})$ cioè A o B ma non (A e B)

Verifichiamo

A	B	A+B	$\overline{AB}$	C
1	1	1	0	0
1	0	1	1	1
0	1	1	1	1
0	0	0	1	0

Costruiamo

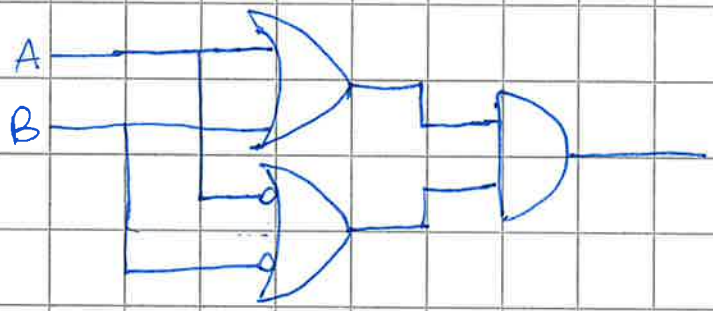


$C = (A+B)(\overline{AB})$

Altra possibilità costruttiva, sempre usando $C = (A+B)(\overline{AB})$

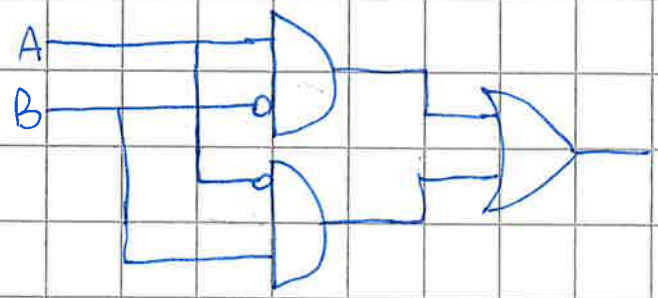
per DeMorgan $= (\overline{A}+\overline{B})$

$C = (A+B)(\overline{A}+\overline{B})$



Ancora un'altra possibilità: nell'OR esclusivo osservo che solo $A \neq B$ è accettato, quindi (A e NON B) o (B e non A)

$C = A\overline{B} + B\overline{A}$



Ce ne sarebbe anche una quarta usando di nuovo DeMorgan...

Mercoledì 14 aprile 2021 ore 12

LOGIC part 2 ch. 16 LEO

Prime applicazioni - "Basic Electronic Logic for Experiments"

- FAN-OUT & FAN-IN (esistono sia lineari sia logici)
  CAEN N625 CAEN N454

Fan-out circuito attivo che permette la distribuzione di un segnale "dividendo" l'ingresso (input) in uno o più segnali identici (stessa ampiezza e forma) in uscita (output)
→ in genere piccoli ritardi
→ da non confondere con splitter passivi (e attivi)

Fan-in accetta più input e fornisce in uscita la loro somma algebrica
→ possono essere bipolari (entrambe le polarità) o unipolari.

Quelli lineari accettano segnali sia analogici sia logici.
Quelli logici solo logici (in questo caso fan-in fa OR)

- RITARDI (Delay Boxes) Es. { CAEN N146 6÷85.5 ns
  *

I segnali si sviluppano temporalmente. Indotta trasmissione, elaborazione, ecc introducono ritardi che posso "sistemare" con cavetti di lunghezza opportuna

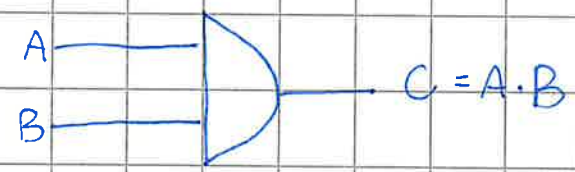
* o Delay Boxes NIM 0-64 ns (passive con cavetti).

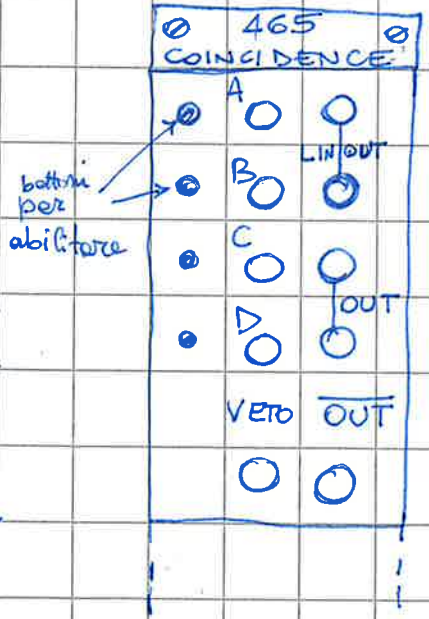
COINCIDENCE UNIT (in lab LECROY 465)

Sono moduli elettronici che determinano se due o più segnali logici sono coincidenti temporalmente e generano in uscita un segnale logico se "vero" e un non segnale se "falso".

Ci sono varie maniere elettroniche di implementare la coincidenza (somma + discriminatore ad es). Quello che interessa è che se c'è un tempo minimo di overlap (resolving time of the coincidence) che dipende dalle larghezze dei segnali si realizza la coincidenza.

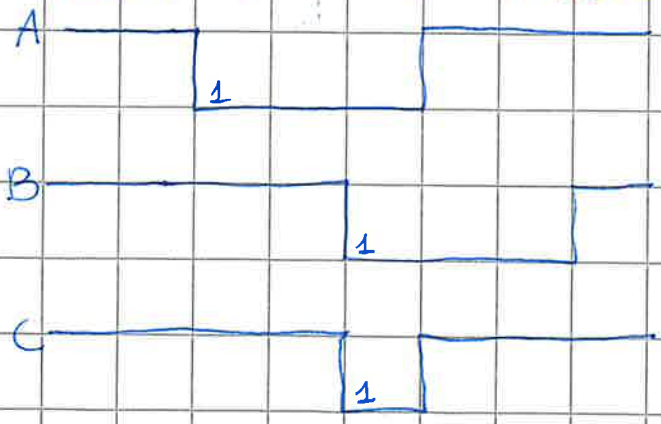
Realizza quindi quanto visto nel logic gate AND





○ input/output LEMO

Esempio per introdurre temporizzazione



In questo esempio A funge da GATE per poter abilitare la coincidenza C
B da il TIMING (tempo) perché arriva dopo A.
o VICEVERSA se A arriva dopo B

Inoltre questo modulo ha altre due sezioni uguali sotto.

MAJORITY LOGIC UNIT

Moduli che realizzano versioni sofisticate e flessibili delle porte logiche viste.
Accettano 1-4 input (o anche di più) e fanno di tutto in output.

FLIP-FLOP (onomatopeico del rumore del relè)

Circuiti sequenziali semplici usati ad esempio come dispositivi di memoria elementare.
Ad esempio nei cronometri quando premo START si memorizza quello stato e si disattiva la possibilità di ripartire ripremendo START accidentalmente (idem per lo STOP).

FLIP-FLOP SR Set-Reset

Circuito a due ingressi S-R e due uscite Q $\bar{Q}$

Il SET=1 manda Q a 1 (e $\bar{Q}$=0) e Q rimane sempre 1 fintanto che non arrivi RESET. Q rimane allora 0 fintanto che non arrivi un SET.

Quando S e R sono entrambi Ø, Q mantiene il valore precedente (memoria), stato neutro del flip-flop.
Non è permessa la combinazione S=1 e R=1

Tabella di verità del flip-flop S-R

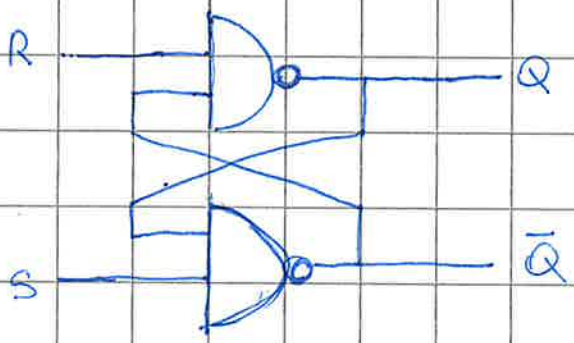
S	R	Q	$\bar{Q}$	Descrizione
0	0	Q_{prec}	$\bar{Q}_{prec}$	Nessuna commutazione (No Change, valore precedente)*
1	0	1	0	SET
0	1	0	1	RESET
1	1	/	/	Combinazione proibita

* memorizza il segnale iniziale, 1-bit memory o LATCH

Combinandone più in parallelo, posso costruire il mondo digitale!

Come posso costruire un flip-flop S-R?

Provo così (SR latch circuit con NAND gates)



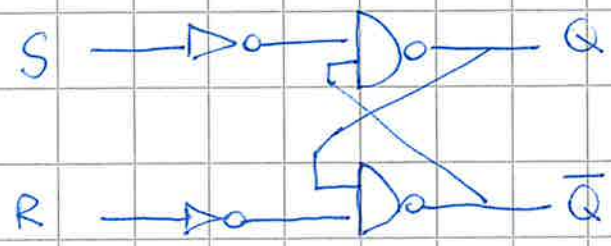
Se parto con R = 0 e S = 0
mi produce la combinazione
proibita Q = 1 e $\bar{Q}$ = 1

Però quando S 0 → 1
Q = 1 e $\bar{Q}$ = 0 ok

Se poi parto R 0 → 1
Q = 1 $\bar{Q}$ = 0 come prima, non cambia

Se infine S 1 → 0
Q = 0 $\bar{Q}$ = 1
Q e $\bar{Q}$ si invertono

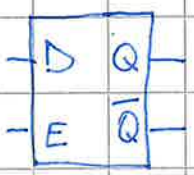
Basta negare S e R e ci sono!



NOTA:
si potevano usare anche NOR

Rimane da sistemare lo stato non permesso:

FLIP-FLOP D (Delay o Data)
o GATED D LATCH



E/C

E = Enable

C = clock

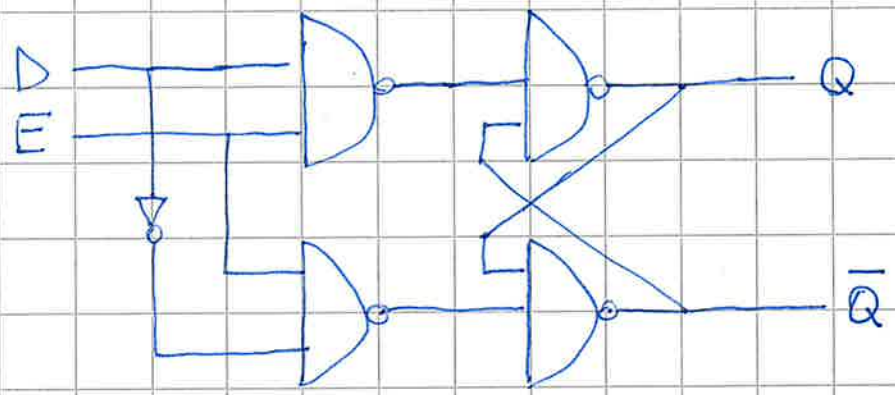


tabella di verità

E/C	D	Q	$\bar{Q}$	Commento
0	X	Q_{prec}	$\bar{Q}_{prec}$	Nessuna variazione
1	0	0	1	Reset
1	1	1	0	Set

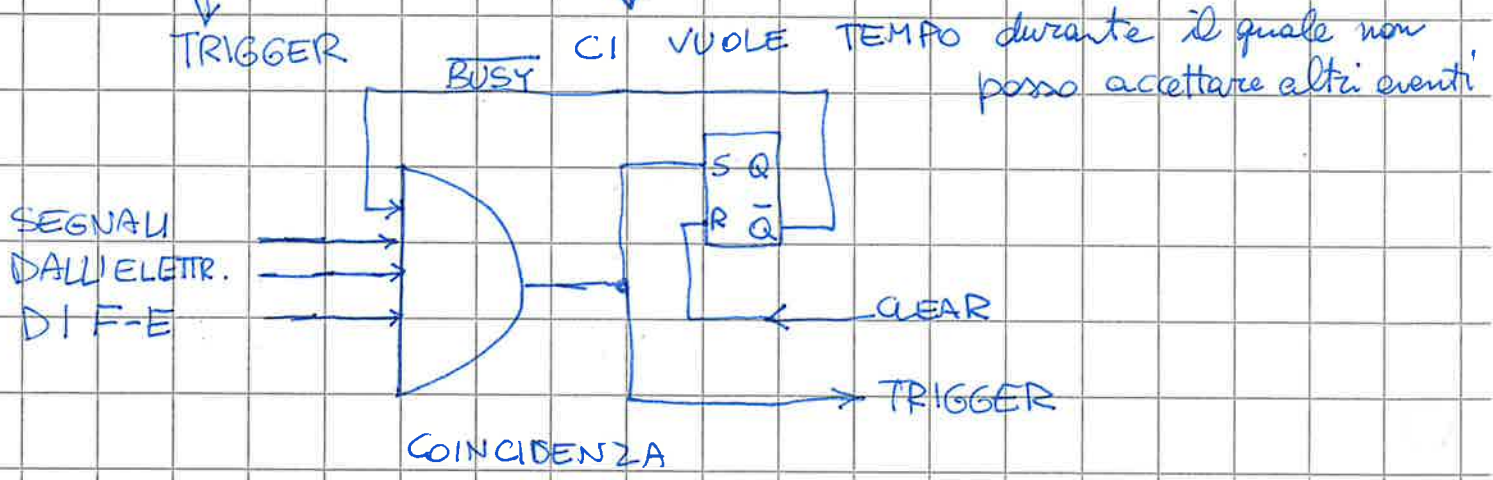
Se E = 1 Q = D

Se E → 0 si ricorda lo stato precedente

Al posto di E ci può essere un clock (es. quello del computer) che apre e chiude E sul fronte di salita. → input di sincronizzazione.
Usato per memori veloci e registri a scorrimento (shift register)

- Inhibit o Busy 16.3

Tornando all'acquisizione e controllo dati: ho un evento che da segnali, li voglio combinare per decidere se siamo bene per poi acquisire.



- Arriva un segnale che è tenuto aperto dal gate AND dalla coincidenza con $\overline{BUSY}$ (non sono BUSY).
- Subito dopo il segnale 1) resetta un flip-flop ⇒ $Q = 1$, $\bar{Q} = 0$ quindi ogni altro evento è bloccato dal gate AND.
  2) Parte il trigger e fintanto che il processamento dei dati non finisce non può arrivare il CLEAR
- Quando finisce, arriva il CLEAR che resetta